A Framework For Financial Botnet Analysis

Marco Riccardi, David Oro and Jesus Luna
eSecurity Research Group
Barcelona Digital Technology Centre
Barcelona, Spain
{mriccardi, doro, jluna}@bdigital.org

Marco Cremonini
Department of Information Technology
University of Milan
Milan, Italy
marco.cremonini@unimi.it

Marc Vilanova
CSIRT
La Caixa
Barcelona, Spain
mvilanova@lacaixa.es

***Abstract*—Financial botnets, those specifically aimed at carrying out financial fraud, represent a well-known threat for banking institutions all around the globe. Unfortunately, these malicious networks are responsible for huge economic losses or for conducting money laundering operations. Contrary to DDoS and spam malware, the stealthy nature of financial botnets requires new techniques and novel research in order to detect, analyze and even to take them down.**

This paper presents a work-in-progress research aimed at creating a system able to mitigate the financial botnet problem. The proposed system is based on a novel architecture that has been validated by one of the biggest savings banks in Spain.

Based on previous experiences with two of the proposed architecture building blocks -the *Dorothy* framework and a blacklist-based IP reputation system-, we show that it is feasible to *map* financial botnet networks and to provide a non-deterministic *score* to its associated zombies. The proposed architecture also promotes intelligence information sharing with involved parties such as law enforcement authorities, ISPs and financial institutions.

Our belief is that these functionalities will prove very useful to fight financial cybercrime.

***Index Terms*—botnets; e-crime forensics framework; honeypots**

I. Introduction

A botnet is a network of compromised computers (also known as *zombies*) remotely controlled and instructed to work in a coordinated fashion by one or more central hosts (known as the *command and control nodes* or C&Cs). A computer turned into a zombie has installed on it tools able to compromise other computers; has remote control mechanisms enabled, and has joined the botnet network by performing coordinated actions. Botnets are responsible for severe Internet threats such as Distributed Denial of Service (DDoS) attacks, spam campaigns and phishing activities. To a great extent the epidemic diffusion of malware has been caused by the spreading activity of botnets as discussed in previous works [1] [2] [3].

Recently, cybercriminals have begun to target online banking using botnets not only for DDoS and spam attacks, but mainly with the purpose of committing financial fraud, like stealing the credentials of customers. A typical example, is the *Zeus* botnet [4] which has compromised around 3.6 million computers only in the US.

Unfortunately, mitigation mechanisms against financial botnets have been severely impaired by some relevant problems:

- Due to the stealthy nature of financial malwares, they are difficult to identify and complex to analyze in an automated way.
- It is difficult and sometimes illegal to obtain the IP addresses of compromised computers due to existing privacy legislation [5] [6].
- Many institutions and organizations are reluctant to share intelligence information about cybercrime cases. This makes it more difficult to coordinate activities among ISPs and law enforcement agencies.
- Currently, there is a lack of supporting frameworks (such as visualization and analysis tools) useful for tracking and taking down financial botnets.

The aim of this paper is to present the architecture and major strategic decisions taken during the design of our supporting framework. The resulting system is based on contributions from our previous research:

1) A heavily customized version of the open source *Dorothy Framework* maintained by the Italian Chapter of the Honeynet Project [7]. The architecture of *Dorothy* [8] is built upon several software modules designed from scratch and aimed at analyzing malware binaries, visually representing their network behaviors and reporting an incident to the involved parties (i.e. ISPs, law enforcement agencies and financial institutions) through a community-oriented online service.
2) A low-latency reputation system [9] based on IP blacklisting. This system is able to provide a scoring mechanism for determining the trustworthiness of a given IP address based on the quality of different blacklists containing that IP address.

Both the *Dorothy Framework* and our IP reputation system have been extended as depicted in Figure 1 with other modules and features in order to cope with the financial botnet phenomenon.

Our belief is that the proposed framework will be able to manage the whole workflow of identifying, analyzing, and mitigating a financial botnet, ranging from an initial malware analysis to the creation of specific feedback and *knowledge* shared with interested parties cooperating to fight cybercrime.

Besides highlighting the importance of tracking and visualizing a botnet's activity, the proposed framework also contributes a methodology to identify malware that targets

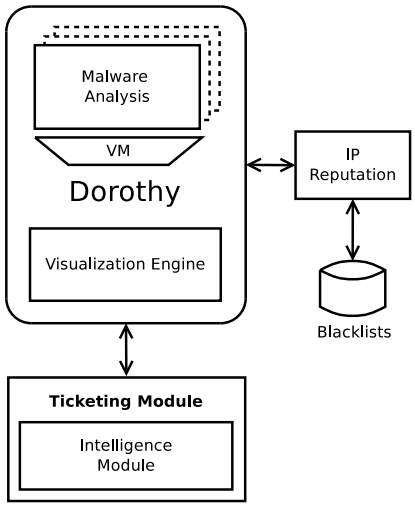

Fig. 1. Overview of the Financial Botnet Framework

financial institutions. It is worth mentioning that very often commercial solutions offered to financial institutions still rely on repositories such as ZeusTracker [10] for gathering information about a botnet, rather than running malware in a controlled environment and analyzing its actual behavior.

The rest of this paper is structured as follows: Sections from II to V describe the architecture of the framework and show the early performance results obtained. Finally, we draw conclusions and discuss our future work in Section VI.

II. Malware Analysis Module

The goal of the Malware Analysis Module (MAM) is to analyze malware and to discover if it affects a specific financial institution. If the analyzed malware represents a threat for a financial institution, then it will be classified as a *banking trojan*. Therefore the malware analysis module could also identify banking trojans that might represent a threat for several financial institutions.

The framework presented in this paper proposes a new architecture for analyzing and classifying malware acting as banking trojans. The rest of this Section explains in more detail the overall architecture.

A. Characterizing the behavior of a banking trojan

Since configuration files of banking trojans are usually encrypted with different symmetric algorithms in different versions of the malware (i.e. this is the case with Zeus), a manual reverse engineering process of the malware must be performed in order to find its respective encryption key.

Inspecting the actual configuration of a banking trojan is useful to verify if it explicitly targets the customers of a specific financial institution by stealing their online banking credentials. For example, a Zeus sample could be defined as a critical banking trojan for the institution X if its configuration file contains a reference to X (i.e. the URL of the online banking site of institution X).

Label	Description
Δt_d	t_3 - t_1
Δt_u	t_5 - t_3
Δt_e	t_5 - t_1

TABLE I
Time periods that characterize a banking trojan

Although this technique may improve the accuracy of analysis by allowing the content of the encrypted communication between the infected host and the C&C to be discovered, it often requires a long time to complete and could be difficult to achieve. For this reason, we propose an alternative method that avoids debugging or reverse engineering for determining if a banking trojan could be reasonably identified as a critical threat for a specific financial institution. The proposed analysis technique has been designed in such a way that it might be executed automatically. A banking trojan behavior could be summarized by the time flow diagram shown in Figure 2.

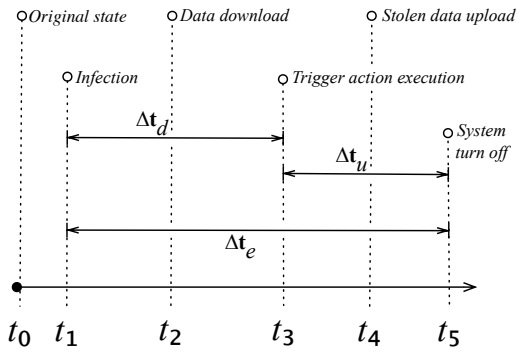

Fig. 2. Banking Trojan Behavior

We define the *zombified timeframe* Δt_e as the interval Δt between the time of the computer infection by the banking trojan and the time of the disconnection of the system (or any other action that makes the system unable to operate as a zombie). During the zombified timeframe Δt_e, two typical phases of a banking trojan can be recognized: the *downloading phase*, whose timeframe is labeled Δt_d and the *uploading phase*, labeled as Δt_u.

The banking trojan under analysis can then be activated in a testing environment by either executing a login operation into a specific online bank or simply when the C&C issues a special command to it. If the malware is activated during the time window Δt_d, it will download the resources needed for its subsequent activities. In our preliminary experiments, we have analyzed the Zeus malware by downloading its configuration files during Δt_d.

Next, in the time window Δt_u the banking trojan will send the stolen data to its pre-configured *drop site* (i.e. a remote site connected to the botnet used for storing data collected by zombies). Table I summarizes the flow with downloading and uploading phases.

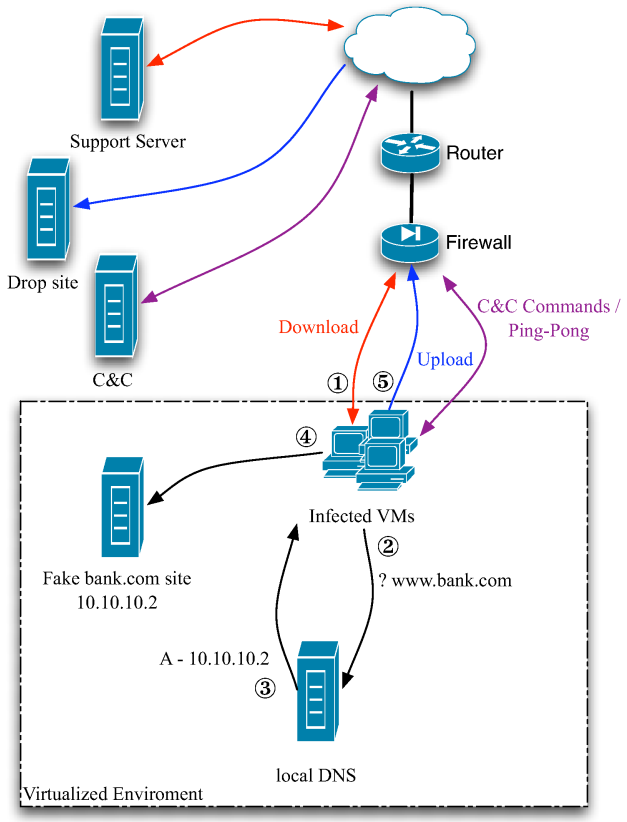

Fig. 3. Malware Analysis Flow

B. Banking trojan analysis process

The overall process for analyzing malware with the purpose of determining if it is a banking trojan is shown in Figure 3 and starts when generic malware is downloaded (step 1 in Figure 3). We propose the use of different virtual machines (with different OS and versions) to execute the malware and to track its network activity using a different instance of a network analyzer for each VM.

Since the VM's local DNS is configured to resolve the real bank's domain name to our local server infrastructure (step 2), all the VMs are then automatically directed to a copy of the original bank web site that has been configured inside the local network (steps 3 and 4).

The malware is then allowed to execute in the controlled environment and, after a preset period of time, the virtual host is stopped and reverted to its original state by loading a saved snapshot. According to the the description of banking trojan behavior given in Subsection II-A, during the Δt_d time window, the infected VM will try to connect to the botnet infrastructure for downloading supporting files (if needed) and, during Δt_u for uploading data stolen from the local copy of the original bank web site (step 5 in Figure 3).

The original *Dorothy Framework* was developed to analyze generic malware by using the workstation version of VMWare [11] virtualization platform. One improvement is the migration to a Kernel-based Virtual Machine. Unlike the standard VMWare solution, the proposed architecture does not run on top of a third-party operating system but is based on a customized Linux Kernel with an integrated hypervisor module. This hypervisor is designed to guarantee high levels of performance for the most resource-intensive applications and thus sharply reduces the total analysis time of the malware analysis module. Preliminary tests have shown that the proposed analysis architecture is capable of analyzing up to 850 malware samples per day.

In the following Subsection we present the *Data Extraction Module* (DEM), which is invoked as the next analysis tool with the purpose of parsing network dump traces obtained during malware execution.

C. Data Extraction Module

Dorothy's Data Extraction Module (DEM) was developed to process the network dump traces recorded during the execution time of each malware sample. In preliminary versions, specific focus was devoted to track IRC-based botnets; however, since most banking trojans use the HTTP/S protocol to communicate with their C&C servers, the original DEM has been modified to detect financial botnet C&Cs, *drop sites*, and Support Delivery Servers (SDSs) communicating over HTTP/S.

According to the banking trojan flow diagram already presented in Figure 2, the DEM identifies a *drop site* by searching for any `POST` request performed during the Δt_u period of time. Some classes of banking trojans use FTP or emails to send stolen user credentials to *drop sites*. To detect them, the customized DEM was also modified to look for these communication mechanisms.

Financial botnet SDSs are identified by filtering all the target hosts reached during the Δt_d period, in particular by searching for HTTP `GET` requests or FTP communications performed by the infected machine. A C&C usually checks the availability of a zombie by exchanging with it an *echo response* verification packet with variable frequency of transmission. However, this communication could be exchanged at any time while the machine is infected and could be encrypted, thus making it difficult to profile a signature based on its content or time frequency. Because of this difficulty, C&C identification is hard to achieve and false positives are likely to occur.

Regarding C&C identification, two strategies have been used. The first one, relies on an observation we made during our preliminary tests that show that a C&C is often co-located with the *drop site* or the SDS component.

Therefore, IP addresses of identified *drop sites* and SDSs may match with those of a C&C.

The second strategy is to consider hosts contacted during Δt_e but which have not already been categorized as a *drop site* or SDS. In this particular case, we will rely on the *IP reputation module* (described in the following Section IV) to figure out their membership of any available blacklist.

At the end of the data extraction process, the results are stored in a relational database that is accessed by the following analysis phases and system modules. The *Dorothy Drone* for example retrieves the C&C information from the database

and mimics the behavior of a real zombie by producing the corresponding replies to all commands received. In this way, it is possible to establish a stateful connection with the C&C and to monitor its activity and behavior, such as issued commands and instructions for downloading binary updates.

Finally, IP addresses associated with botnet components found by the DEM are also used by the IP reputation system to feed its own database. The whole collected data can also be included into a new incident report created via the newly developed *Ticketing System Module* presented in Section V.

III. Visualization Module

Previous research with Dorothy [8] proved that displaying information about the activity of a botnet through link graphs and geographic maps is a powerful technique to quickly understand and share knowledge among all the parties involved in botnet takedown actions.

The framework proposed in this paper relies on a customized version of Dorothy's Data Visualization Module (DVM) to visualize the information gathered by previous modules. The goal of the newly customized DVM is to offer a global picture that summarizes all the information needed to investigate a financial botnet case (i.e. by law enforcement agencies). The visualization process has been modeled according to methodologies described in previous works [12].

For the purpose of this research, we have designed the visualization engine based on the following questions:

- *Who* has been contacted by a zombie machine?
- *What* communication flow occurred between hosts that are members of a botnet?
- *What* has been downloaded or uploaded?

Let us introduce some examples of how advanced visualization features may enhance the analysis and management of a botnet threat:

- By highlighting the target of HTTP `GET`/`POST` requests performed by the infected machine during Δt_u and Δt_d, it might be possible to intuitively identify both an SDS and a *drop site*.
- Based on our practical experience with financial CSIRT[1] operations, we have concluded how visualizing information about the size of the exchanged data could help the analyst to identify suspicious communications and focus on them for further investigations.
- By showing all the hosts contacted during Δt_u we can also determine which protocols were used by the infected machine to send stolen data.

The DVM uses the criteria shown in Table II for identifying the different entities related to a financial botnet.

Finally, the rest of this section presents visualization techniques implemented in the proposed DVM architecture.

[1]Computer Security Incident Response Team

Entity	Detection Method	Time
C&C	Blacklist Matching	Δt_e
SDS	HTTP `GET`/FTP Parsing	Δt_d
Drop site	HTTP `POST`/FTP/SMTP	Δt_u

TABLE II
Detection methods used for entity categorization

A. Link graph

When the DVM accesses data stored in the database through the MAM (as explained in Section II), it also perfoms parsing and filtering operations with the purpose of generating a *3-tuple* of the form *(source, service, target)*.

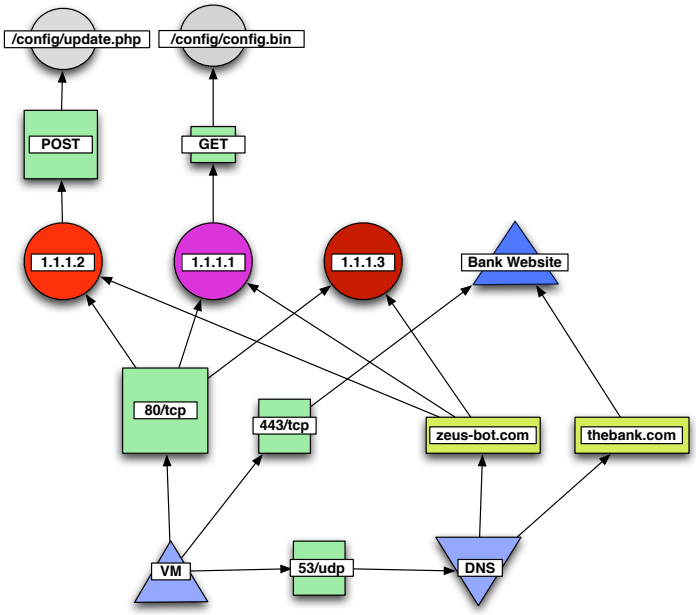

Fig. 4. Link Graph

The graph depicted in Figure 4 is built using visual transformations from Table III. Size of nodes are proportional to the number of corresponding entries in the data source. So the more entries are found, the larger the node size is. The generated link graph summarizes all the connections established by the financial zombie under analysis.

Entity	Color	Shape
C&C	Purple Red	Circle
Fake Bank Server	Dark Blue	Triangle
Drop Zone	Red	Circle
SDS	Purple	Circle
Services	Green	Square
Infected VM	Light Blue	Triangle
Local DNS	Light Blue	Inverted Triangle

TABLE III
Visual transformation graph properties

In our preliminary tests, we downloaded several banking trojans, analyzed all the traffic generated by their respective zombies using our framework and categorized each zombie

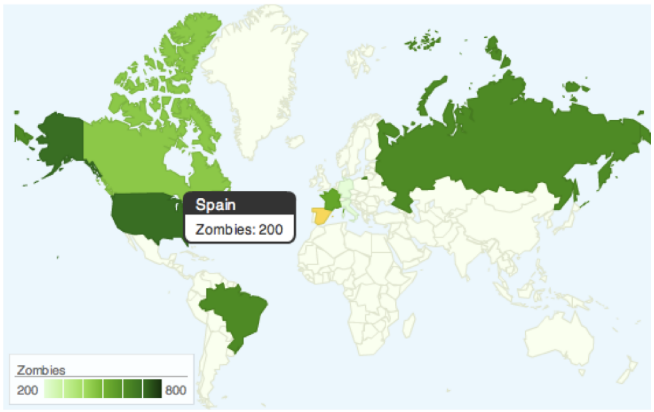

Fig. 5. Heat Map

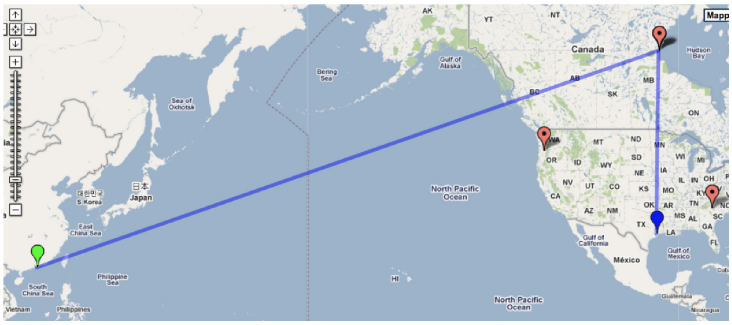

Fig. 6. Geographical Map

according to the C&C being contacted. In this way, it was possible to merge traffic dumps from zombies that were referring to the same C&C, so the DVM could process it. The overall outcome of this activity was a graph of the analyzed botnet, useful in investigation processes related to a particular financial botnet architecture.

B. Heat maps

In order to visualize the huge amount of IP addresses gathered by the IP Reputation Module (see Section IV), we focused our efforts on providing a method for displaying information that is geographically dispersed (i.e. using *heat maps*). A typical *heat map* generated by our framework is depicted in Figure 5.

C. Map graph

The original Dorothy Web GUI was customized to allow a financial CSIRT to easily visualize financial botnets that are targeting its customers. Our belief is that visualizing C&Cs, SDSs and *drop sites* through a geographical map could be an intuitive and immediate way to show how the financial botnet network is distributed, which may help in the analysis (as shown in Figure 6).

For instance, by showing the C&Cs geographical distribution, a bank's CSIRT could be helped in defining a strategy aimed at stopping the botnet's activity by considering that different countries have different laws about cybercrime. The geographical representation could also suggest possible geopolitical motivations in operating a financial botnet, apart from traditional online fraud.

In addition, the interactive map we have contributed offers the possibility to represent data through different visualization layers by displaying the information about a selected C&C such as its own link graph (Section III-A).

Finally, apart from Dorothy Web GUI original panels (*General*, *Charts*, and *Malware*), a *Status Panel* was added to show information regarding the Ticketing System Module such as the first notification that was sent, the recipient of a particular notification and the status of the related ticket. The Ticketing System Module will be further described in Section V.

IV. IP Reputation Module

In previous research [9] we proved that it might be possible to discover IP addresses related to a financial botnet, by combining the information coming from well-known IP blacklists providers. In order to do so, we developed a set of metrics (i.e. infection latency) that are able to quantify the *quality* of a particular IP blacklist (score or *reputation*). Formally speaking:

- Let B be a set of all trusted blacklist providers and M a set of bitmaps.
- Let M_i encode whether a specific IP address was found inside the blacklist B_i $(0 \leq i \leq |B|)$ or not.

$$M_i = \begin{cases} 1 & \text{if } IP \in B_i \\ 0 & \text{otherwise} \end{cases}$$

- Let $\delta(IP)$ be a reputation function defined as:

$$\delta(IP) = \begin{cases} \{\varnothing\} & \text{if } \nexists B_i \in B : IP \in B_i \\ \{M\} & \text{otherwise} \end{cases}$$

With the δ function we obtain the reputation of a given IP address. In summary, the more bits are set as "1" in a bitmap, the more likely it is that an IP address corresponds to a zombie host that is a member of a financial botnet.

The following new features are required by the original IP reputation system in order to use it within the framework proposed in this paper:

- **Storage.** All the collected information needs to be summarized inside an AVL tree [13], then it is indexed by IP address and finally it is stored inside nodes with its corresponding bitmap M. For our first implementation (see Figure 7), each node was populated with a 40-bit tuple $\langle IP, M \rangle$ due to the fact that we were only fetching IP blacklists from 8 different providers. Since IPv4 addresses are only 32 bits long, it was possible to store the whole IP address space (up to 4 billion addresses) in RAM memory with a minimal effort (8 GB of RAM or even less).
- **Performance.** The proposed framework should be able to feed in near real-time anti-fraud systems of financial institutions. For this reason, we decided not to rely on a RDBMS for storing IP addresses from malicious hosts. As shown later in this section, without performing disk

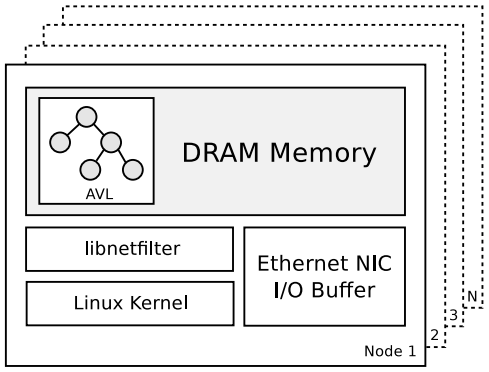

Fig. 7. IP Reputation System Architecture

accesses low-latency is guaranteed at $(\Theta \log n)$ cost in the worst-case scenario –lookup operations–.

- **Security.** The reputation system proposed in [9] needs to be further extended in order to meet some minimum trust and security guarantees. If it is meant to be used outside financial institution boundaries (i.e. by an external ticketing system), then the IP reputation system responses should be digitally signed by a trusted third party (just as happens with protocols such as OCSP [14]).
- **Detection rate.** It is important to invest more research in order to improve the IP reputation system detection rate. As mentioned in [9], future work should take into account better reputation algorithms along with the effect of dynamic IP addresses –DHCP–.

The above-mentioned features allow us to use the IP reputation system with other modules of the proposed framework (i.e. to feed the Ticketing System Module). As a result of this, ISPs could be able to apply long-term filters to IP addresses with bad reputation. As has been mentioned in Section II, the IP reputation system would be able to receive and send information to the Malware Analysis Module (see Section II).

So far our research has demonstrated the performance of our first IP reputation system implementation. As expected, RAM memory is extremely unlikely to be accessed and for this reason if we discard overhead latencies such as I/O interrupt handling or the code parsing performed inside the Linux Kernel, the average obtained lookup latency is roughly 8μs. If we consider workloads that are less than 50 million IP addresses, it is possible to achieve a 99% hit rate in caches. In order to prove this hypothesis, we have made measurements with the OProfile [15] Linux Kernel module and the CPU performance counters. Table IV shows the obtained results for a 50 million IPv4 address workload.

V. Ticketing System Module

Once a financial botnet has been automatically discovered by the analysis performed during the Malware Analysis Module (see Section II), it is necessary to send notifications to all the involved parties such as banks, ISPs, ccTLD registrars, antivirus software companies and law enforcement agencies.

	Total Accesses	Read	Write
DCache Refs.	50.9M	36.2M	14.7M
DCache Misses	0.52M	0.39M	0.13M
L2 Misses	0.24M	0.12M	0.12M
DCache Miss Rate	1%	1%	0.8%
L2 Miss Rate	0.4%	0.3%	0.1%
L2 Refs.	0.53M	0.4M	0.13M
L2 Misses	0.25M	0.13M	0.12M
L2 Miss Rate	0.1%	0%	0.8%

TABLE IV
Cache behavior for 50 million IP addresses

Since each entity has a different role, the only way to be successful in neutralizing botnet infrastructures and arresting the criminals who are running them is by promoting an efficient coordination among the different authorities. For this purpose, an email-based notification system has been developed inside the Ticketing System Module (TSM).

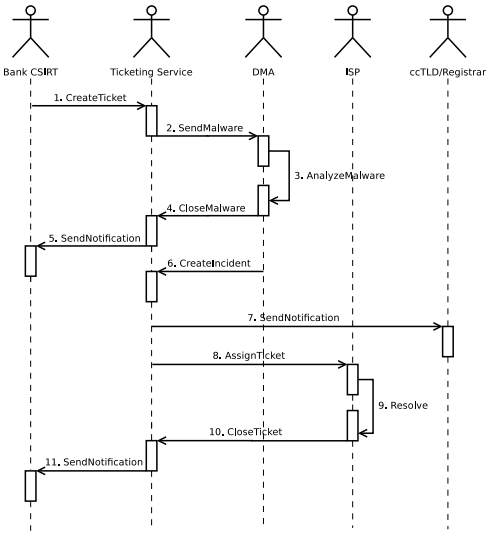

Fig. 8. Ticket Workflow

As depicted in Figure 8, the bank's CSIRT starts the workflow by requesting a new malware analysis from the MAM (see Section II). Malware samples sent by e-mail are then automatically analyzed (steps 2 and 3 in Figure 8) and a preliminary report is attached for later revision. This report is encoded in XML [16] following the rules defined by the IEEE-SA Industry Connections Malware Working Group (ICSG).

Once all the intelligence information has been gathered, a new ticket is created and returned to the bank's CSIRT (step 5 in Figure 8).

If the malware analysis process has identified a banking trojan as a real threat for several financial institutions (with the methods described in II-A), then another set of tickets is created (a newer one for each one of the newly discovered C&C IP addresses and domains).

All tickets generated by the TSM after a C&C IP address has been discovered, are sent to the security providers of financial institutions who are in charge of mitigating financial malware

(steps 8-10 in Figure 8). This is a realistic assumption taking into account our practical experience with financial CSIRTs.

In fact, we have learnt that in most cases, such *security providers* are broadband and telecommunications companies which are able to adopt an appropriate mitigation strategy (i.e. *sinkholing* of botnet domains and *blacklisting* of C&C's IP addresses).

Since most of the DSL/Cable/FTTH equipment deployed at subscribers' homes are often configured by default with DNS and gateway IP addresses provided by the broadband companies, it is possible to quickly mitigate the impact of a financial botnet just by blocking malicious IP addresses and domains at the ISP level (steps 8-10 in Figure 8).

In parallel with this action, a notification email is automatically sent to the affected ccTLD just by performing a query at `WHOIS` servers with the purpose of obtaining contact email addresses from the `Abuse` records (step 7).

Ensuring privacy when exchanging sensitive information between financial instutions, the different law enforcement agencies and ISPs have also been considered during the design phase of the Ticketing System Module. All emails sent by the TSM (with the expection of those sent to untrusted ccTLDs) are encrypted and signed using public key cryptography (OpenPGP [17]).

Finally, when an incident case has been solved (steps 10-11 in Figure 8), closed tickets are stored in a database with the purpose of building a solid knowledgebase for further investigations and for producing statistics of the amount of time employed in neutralizing botnet infrastructures.

At this point, all collected information can be used by the law enforcement authorities for starting legal actions against cybercriminals.

VI. Conclusions and Future Work

In this paper we have introduced a novel framework to detect, visualize and in general to share intelligence about financial botnets. Such a system would greatly help to mitigate this threat by sharing information among different parties, ranging from financial institutions' antifraud systems (IP Reputation Module) to law enforcement authorities (i.e. via the Visualization Module). This research contributed a new way to automatically analyze generic malware and to categorize it as a banking trojan.

Despite this being a work-in-progress research, our preliminary implementation of the proposed modules proves the usefulness of these tools for mitigating the threat posed by financial botnets. Due to their stealthy nature and uncommon behavior, financial botnets should be addressed differently from traditional botnets characterized by DDoS and spam activity.

Our future work will be focused on adding new features to the system such as improving the different mechanisms used for detecting compromised computers through new reputation/scoring algorithms and fusing information provided by both ISPs and blacklist providers.

In addition, the Ticketing Module will add support for the new Instant Object Description Exchange Format (IODEF) [18] extensions, recently released with the purpose of defining a XML standard for reporting *e-crime* incidents [19].

We are also exploring new technologies that might allow us to deploy this system in such a way that intelligence sharing may be facilitated. In particular, we are referring to cloud-based implementations whose scalability and dynamics may let this system meet higher performance requirements that will enable graphical representations of large botnets.

Finally, future enhancements of the Ticketing Module, will involve the use of text-to-speech (TTS) technology for performing automated voice calls to both Domain Registrars and ccTLDs with the phone numbers retrieved from the `WHOIS` records.

Acknowledgements

This work has been financially supported by La Caixa savings bank. We also would like to thank Richard Hayden, Toni Felguera and David Hernando for their insightful comments and suggestions during the development of the framework.

References

[1] E. Cooke, F. Jahanian, and D. McPherson, "The zombie roundup: Understanding, detecting, and disrupting botnets," in *Proceedings of the USENIX SRUTI Workshop*, 2005, pp. 39–44.

[2] M. Abu Rajab, J. Zarfoss, F. Monrose, and A. Terzis, "A multifaceted approach to understanding the botnet phenomenon," in *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*. ACM, 2006, p. 52.

[3] P. Bacher, T. Holz, M. Kotter, and G. Wicherski, "Know your enemy: Tracking botnets," *The Honeynet Project*, 2005.

[4] H. Binsalleeh, T. Ormerod, A. Boukhtouta, P. Sinha, A. Youssef, M. Debbabi, and L. Wang, "On the Analysis of the Zeus Botnet Crimeware Toolkit," in *Proceedings of the 8th Annual Conference on Privacy, Security and Trust*. IEEE, 2010, pp. 31–38.

[5] D. Baumer, J. Earp, and J. Poindexter, "Internet privacy law: A comparison between the United States and the European Union," *Computers & Security*, vol. 23, no. 5, pp. 400–412, 2004.

[6] C. Kuner, *European data protection law*. Oxford University Press.

[7] L. Spitzner, "The honeynet project: Trapping the hackers," *IEEE Security and Privacy*, pp. 15–23, 2003.

[8] M. Cremonini and M. Riccardi, "The Dorothy Project: An Open Botnet Analysis Framework for Automatic Tracking and Activity Visualization," in *Proceedings of the 5th European Conference on Computer Network Defense (EC2ND)*. IEEE, 2010, pp. 52–54.

[9] D. Oro, J. Luna, T. Felguera, M. Vilanova, and J. Serna, "Benchmarking IP blacklists for financial botnet detection," in *Proceedings of the 6th International Conference on Information Assurance*. IEEE, 2010, pp. 98–103.

[10] "Zeus Tracker," *https://zeustracker.abuse.ch*, 2010.

[11] B. Walters, "VMware virtual platform," *Linux Journal*, vol. 1999, no. 63es, p. 6, 1999.

[12] R. Marty, "Applied security visualization," 2008.

[13] C. Ellis, "Concurrent search and insertion in AVL trees," *IEEE Transactions on Computers*, vol. 100, no. 29, pp. 811–817, 1980.

[14] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, "RFC2560: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol (OCSP)," *Internet Engineering Task Force (IETF)*, 1999.

[15] W. Cohen, "Multiple Architecture Characterization of the Linux Build Process with OProfile," in *Workshop on Workload Characterization*, 2003.

[16] "IEEE Standards Association: Industry Connections Malware Working Group," *http://standards.ieee.org/prod-serv/indconn/icsgmal/index.html*, 2010.

[17] J. Callas, L. Donnerhacke, H. Finney, and R. Thayer, "RFC 2440: OpenPGP Message Format," *Internet Engineering Task Force (IETF)*, 1998.
[18] R. Danyliw, J. Meijer, and Y. Demchenko, "RFC 5070: The Incident Object Description Exchange Format," *Internet Engineering Task Force (IETF)*, 2007.
[19] P. Cain and D. Jevans, "RFC 5901: Extensions to the IODEF-Document Class for Reporting Phishing," *Internet Engineering Task Force (IETF)*, 2010.